Fast Eigenspace Approximation using Random Signals

Johan Paratte and Lionel Martin *†

November 4, 2016

Abstract

We focus in this work on the estimation of the first k eigenvectors of any graph Laplacian using filtering of Gaussian random signals. We prove that we only need k such signals to be able to exactly recover as many of the smallest eigenvectors, regardless of the number of nodes in the graph. In addition, we address key issues in implementing the theoretical concepts in practice using accurate approximated methods. We also propose fast algorithms both for eigenspace approximation and for the determination of the kth smallest eigenvalue λ_k. The latter proves to be extremely efficient under the assumption of locally uniform distribution of the eigenvalue over the spectrum. Finally, we present experiments which show the validity of our method in practice and compare it to state-of-the-art methods for clustering and visualization both on synthetic small-scale datasets and larger real-world problems of millions of nodes. We show that our method allows a better scaling with the number of nodes than all previous methods while achieving an almost perfect reconstruction of the eigenspace formed by the first k eigenvectors.

Keywords— Graph signal processing, low-rank reconstruction, partitionning, spectral graph theory, spectrum analysis, subspace approximation, visualization

1 Introduction

Although the questions related to data analytics such as clustering or visualization have received a lot of attention in the past decades, their study is also gaining importance due to the amount of data that one would like to treat nowadays. In particular, this current trend requires that methods must be able to accommodate with large data sets. This imposes two important constraints in the design of new techniques: one must ensure that the complexity and the storage required to process the data are as low as possible.

In the past, many accurate techniques have been introduced to tackle the questions of dimensionality reduction, clustering, and visualization. Mostly, they used the fact shared among those problems that high-dimensional data (in $\mathbb{R}^N$) often admits an accurate low-dimensional intrinsic representation. Finding this embedding alleviate the processing and storage constraints of further processing tasks by representing data points in a space of smaller dimension $d \ll N$.

Eigendecomposition has been at the core of famous techniques used to extract low-dimensional embeddings from high-dimensional data by using the eigenvectors associated with specific eigenvalues. This has been used for partitioning (e.g., spectral clustering [1, 2]), data visualization (e.g., Laplacian eigenmaps [3]), but also simply as a dimensionality reduction

*EPFL, Ecole Polytechnique Fédérale de Lausanne, LTS2 Laboratoire de traitement du signal, CH-1015 Lausanne, Switzerland

†The authors contributed equally.

technique for preprocessing (e.g., principal components analysis [4]). Alternatively, stochastic algorithms for dimensionality reduction (e.g., stochastic neighbor embedding (SNE) [5]) appeared as interesting alternatives, especially for visualization. The main drawback of all the aforementioned techniques is that they tend not to scale well as they have a rather high complexity (e.g., a partial eigendecomposition being $\mathcal{O}(kN^2)$ while SNE is $\mathcal{O}(N^2)$).

The classical way to recover the eigenspace of a symmetric matrix $\mathcal{L}$ is to diagonalize it as $\mathcal{L} = \mathbf{U}\Lambda\mathbf{U}^*$, with $\mathbf{U}$ being the matrix of eigenvectors and Λ the matrix of eigenvalues, and take the first k columns of $\mathbf{U}$. The diagonalization is typically done using a singular value decomposition (SVD) of a symmetric matrix of size N is $\mathcal{O}(N^3)$ which is intractable even for medium scale N. A great deal of work has been done on faster ways to compute eigenvalues and eigenvectors of $\mathcal{L}$ efficiently (see [6] for a review). The fastest methods are variants of Arnoldi or Lanczos iteration methods ([7] and [8] respectively) such as Implicitly Restarted Arnoldi Method (IRAM) [9] or Implicitly Restarted Lanczos Method (IRLM) [10]. The preferred method for graph Laplacians is the IRLM since the matrix is symmetric and sparse most of the time. The IRLM has a worst case complexity of $\mathcal{O}(h(|\mathcal{E}|k + k^2N + k^3))$, with h the number of iterations to reach convergence and assuming there are $\mathcal{O}(k)$ extra Lanczos steps [6]. If we consider sparse graphs with $|\mathcal{E}| \approx \mathcal{O}(N)$ and a fixed k independent of the value of N, the complexity of the IRLM is bounded by the term $\mathcal{O}(k^2N)$.

Since the exact computation of the eigenspace proves to be expensive, several angles were considered to approximate the result. Physicists came with a solution to the problem of eigenspace determination using contour integration techniques for the reduction of the matrices on which to apply the eigendecomposition [11], that allows improving the complexity with almost no loss of precision. Meanwhile, with the additional constraint that the matrix should contain a subset of the columns of the original matrix, Boutsidis et al. [12] propose a fast method to approximate low-rank matrix reconstruction (whose optimal solution is the eigenspace generated by the first eigenvectors). Some works, such as [13], focus on their side, on the determination of the first non-trivial eigenvector only. Finally, Bai [14] proposes a solution for the approximation of eigenvectors using tridiagonalization of sparse matrices that requires "efficiently" sparse matrices as input. Although this might not necessarily apply in practice depending on the data set at hand, it proved to be efficient in various problems involving modeling physical phenomena with strong locality properties.

Instead of computing the eigenspace as features of the data points in the new space, distance preservation can be considered sufficient depending on the application. Indeed, for tasks such as clustering, supposing an algorithm such as k-means is performed as the final assignment step, the preprocessing for dimensionality reduction only requires pairwise distances between points to be preserved in the new space. In this mind, [15] presents a clustering algorithm that avoids the computation of an SVD by computing polynomial approximations and using the Johnson-Linderstrauss lemma.

On the same line, the authors of [16] show that the power method (computing powers of the normalized weight matrix) gives a good approximation of the eigenvectors for distance preservation. They give a bound on the power required to obtain a good approximation of the clustering. This is among the first works, to our knowledge, to use random signal multiplied by powers of the weight matrix.

Even more recently, [17] proposed a fast algorithm for graph clustering which is provably as good as spectral clustering. The first half of their work uses random signal filtering and provides a result similar to the one presented in [16]. Moreover, they additionally show that only a subset of the nodes must be assigned with k-means and that the rest can be inferred from the graph structure by solving an optimization problem. They state bounds on the number of signals required and the number of nodes to label with k-means.

In this work, we present a new algorithm for Fast Eigenspace Approximation using Random Signals (FEARS) to estimate the first k eigenvectors using random signal filtering techniques that were already used in the works on distance preservation. This time, however, we do not simply find a mapping for distance preservation but we are able to obtain the partial

eigenspace, with a total complexity inferior to the previous works.

In this context, our paper proposes various improvements to the field, whose main contributions are:

- a very efficient scheme for the estimation of eigenspaces using filtering of random graph signals
- a proven tight bound for the number of random signals needed for perfect recovery
- algorithms and implementations with practical considerations regarding filter design, fast filtering, and numerical stability
- an accelerated method for the count of eigenvalues in a given range

The paper is organized as follows. In Section 2, we recall the fundamentals of graph signal processing and define the notation. Section 3 develops the main results of this paper from the theoretical point of view while Section 4 presents its applied counterpart and also presents the algorithms for fast spectral embedding and eigencount estimation. Later in Section 5, we show the validity and benefits of our method and compare with the state of the art through several experiments. Finally, Section 6 proposes interesting open problems in the domain as well as potential future work to address.

2 Background

Graph nomenclature Let us define $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ as an undirected weighted graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges representing connections between nodes in $\mathcal{V}$. The vertices $v \in \mathcal{V}$ of the graph are ordered from 1 to $N = |\mathcal{V}|$. The matrix $\mathbf{W}$, which is symmetric and positive, is called the weighted adjacency matrix of the graph $\mathcal{G}$. The weight $\mathbf{W}_{ij}$ represents the weight of the edge between vertices v_i and v_j and a value of 0 means that the two vertices are not connected. The degree $d(i)$ of a node v_i is defined as the sum of the weights of all its edges $d(i) = \sum_{j=1}^{N} \mathbf{W}_{ij}$. Finally, a graph signal is defined as a vector of scalar values over the set of vertices $\mathcal{V}$ where the i-th component of the vector is the value of the signal at vertex v_i.

Spectral theory The combinatorial Laplacian operator $\mathcal{L}$ can be defined from the weighted adjacency matrix as $\mathcal{L} = \mathbf{D} - \mathbf{W}$ with $\mathbf{D}$ being the degree matrix defined as a diagonal matrix with $\mathbf{D}_{ii} = d(i)$. One alternative and often used Laplacian definition is that of the normalized Laplacian $\mathcal{L}_n = \mathbf{D}^{-\frac{1}{2}}\mathcal{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{\frac{1}{2}}$. Since the weight matrix $\mathbf{W}$ is symmetric positive semi-definite, so is $\mathcal{L}$ by construction. By application of the spectral theorem, we know that $\mathcal{L}$ can be decomposed into an orthonormal basis of eigenvectors noted $\{\mathbf{u}_\ell\}_{\ell=0,1,\dots,N-1}$. The ordering of the eigenvectors is given by the eigenvalues noted $\{\lambda_\ell\}_{\ell=0,1,\dots,N-1}$ sorted in ascending order $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{N-1} = \lambda_{\max}$. In a matrix form we can write this decomposition as $\mathcal{L} = \mathbf{U}\Lambda\mathbf{U}^*$ with $\mathbf{U} = (\mathbf{u}_1|\mathbf{u}_2|\dots|\mathbf{u}_{N-1})$ the matrix of eigenvectors and Λ the diagonal matrix containing the eigenvalues in ascending order. Given a graph signal f, its graph Fourier transform is thus defined as $\hat{f} = \mathcal{F}(f) = \mathbf{U}^* f$, and the inverse transform $f = \mathcal{F}^{-1}(\hat{f}) = \mathbf{U}\hat{f}$. It is called a Fourier transform by analogy to the continuous Laplacian whose spectral componants are Fourier modes, and the matrix $\mathbf{U}$ is sometimes referred to as the graph Fourier matrix (see e.g., [18]). By the same analogy, the set $\{\sqrt{\lambda_\ell}\}_{\ell=0,1,\dots,N-1}$ is often seen as the set of graph frequencies [19].

Graph filtering In traditional signal processing, filtering can be carried out by a pointwise multiplication in Fourier. Thus, since the graph Fourier transform is defined, it is natural to consider a filtering operation on the graph using a multiplication in the graph

Fourier domain. To this end, we define a graph filter as a continuous fonction $g : \mathbb{R}_+ \to \mathbb{R}$ directly in the graph Fourier domain. If we consider the filtering of a signal f, whose graph Fourier transform is written $\hat{f}$, by a filter g the operation in the spectral domain is a simple multiplication $\hat{f}'[\ell] = g(\lambda_\ell) \cdot \hat{f}[\ell]$, with f' and $\hat{f}'$ the filtered signal and its graph Fourier transform respectively. Using the graph Fourier matrix to recover the vertex-based signals we get the explicit matrix formulation for graph filtering:

$$f' = \mathbf{U} g(\Lambda) \mathbf{U}^* f,$$

where $g(\Lambda) = \mathrm{diag}(g(\lambda_0), g(\lambda_1), \ldots, g(\lambda_{N-1}))$. The graph filtering operator $g(\mathcal{L}) := \mathbf{U} g(\Lambda) \mathbf{U}^*$ is often used to reformulate the graph filtering equation as a simple vector-matrix operation $f' = g(\mathcal{L}) f$.

Since the filtering equation defined above involves the full set of eigenvectors $\mathbf{U}$, it implies the diagonalization of the Laplacian $\mathcal{L}$ which is costly for large graphs. To circumvent this problem, one can represent the filter g as a polynomial approximation, since polynomial filtering only involves the multiplication of the signal by a power of $\mathcal{L}$ of the same order as the polynomial. Filtering using good polynomial approximations can be done using Chebyshev or Lanczos polynomials [20, 21].

3 Eigenspace estimation using random signals

The goal of our method is to get the best estimation of the subspace of the graph Laplacian $\mathcal{L}$, denoted $\mathbf{U}_k$, for the lowest computational cost. In a similar approach to [17] and [16], we consider the filtering of random signals. We chose an ideal low-pass filter $g(\mathcal{L}) = \mathbf{U}_k \mathbf{U}_k^*$ to achieve this goal. Throughout this section, we prove the following theorem, one of our main results:

Theorem 1. *Let g be an ideal low-pass filter of cutoff frequency λ_k, let $\mathbf{R} \in \mathbb{R}^{N \times d}$ a random matrix formed of entry-wise independent and identically distributed Gaussian random variables $\sim \mathcal{N}(0, \frac{1}{d})$. Let $\mathcal{L}$ be the Laplacian of any graph $\mathcal{G}$.*

For any $d \geq k$, performing a QR decomposition on the result of the filtering of $\mathbf{R}$ by g provides the first k eigenvectors of $\mathcal{L}$ altered only by a rotation in $\mathbb{R}^k$.

3.1 Exact eigenspace recovery with random signals

Assuming we pack d Gaussian random signals with i.i.d. entries $\sim \mathcal{N}(0, \frac{1}{d})$ in a Gaussian random matrix $\mathbf{R} \in \mathbb{R}^{N \times d}$, the result of the filtering using the filter g can be written as $\mathbf{M} = \mathbf{U}_k \mathbf{U}_k^* \mathbf{R} = \mathbf{U}_k \mathbf{R}_k$. We will first state a result regarding $\mathbf{R}_k$ and then use $\mathbf{R}_k$ directly to compute the projection.

Lemma 1. *Let $\mathbf{U}$ be an orthonormal basis and denote $\mathbf{U}_k$ a subset of k of its rows.*

The projection of a Gaussian random matrix $\mathbf{R} \sim \mathcal{N}(0, \sigma^2 I)$ onto $\mathbf{U}_k$ preserves all the Gaussian properties of $\mathbf{R}$.

Proof. The multiplication of a Gaussian random matrix by a basis such as $\mathbf{U}$ preserves all the properties of the initial random matrix (Gaussian, entry-wise independence, identical mean, variance, and size). This proof can be found in the appendix of this paper.

Selecting any subset of the rows of $\mathbf{U}$ changes the size but conserves the orthonormal properties over the rows. Indeed, without loss of generality on the rows selection, we have

$$\begin{pmatrix} I_k \\ 0 \end{pmatrix} \mathbf{U}\mathbf{R} = \begin{pmatrix} I_k \\ 0 \end{pmatrix} \mathbf{R}' = \mathbf{R}_k \tag{1}$$

Thus, only the size will be altered compared to a multiplication by the full matrix $\mathbf{U}$. This concludes the proof. □

With lemma 1, we have that $\mathbf{R}_k \in \mathbb{R}^{k\times d}$ is i.i.d. Gaussian of zero mean and variance $\frac{1}{d}$. The next step is to show that $\mathbf{R}_k$ is full rank.

Lemma 2. *Let $\mathbf{R}_k \in \mathbb{R}^{k\times d}, d \geq k$ be a Gaussian random matrix of entry-wise i.i.d. $\sim \mathcal{N}(0,\sigma^2)$.*

R_k is a full rank matrix with probability 1. That is, $\text{rank}\{\mathbf{R}_k\} = k$ *since $d \geq k$.*

Proof. Let us consider the limit case $d = k$. In this case we have to show that the square $(k \times k)$ matrix $\mathbf{R}_k$ is non-singular. Indeed, the set of singular Gaussian random matrices $\mathcal{R}_s = \{\mathbf{R}_k : \det(\mathbf{R}_k) = 0\}$ is of dimension $k-1$ since it is generated by the zeros of a polynomial of order k. Moreover, since the complete set $\mathcal{R} = \{\mathbf{R}_k\}$ has dimension k, the codimension of $\mathcal{R}_s$ is 1. Thus, the set $\mathcal{R}_s$ is a null set, which means that picking a matrix at random from the set $\mathcal{R}$ returns a matrix from $\mathcal{R}_s$ with probability 0. Hence, $\mathbf{R}_k$ is non-singular with probability 1.

If we consider $d > k$, any square matrix formed of k of the columns of $\mathbf{R}_k$ has rank k following the proof above for the square case. Now, adding columns to this matrix can not change the rank since it can not reduce it and the matrix is full rank already. □

As lemma 2 is critical to the proof, we make a slight digression and discuss its numerical approximation. Indeed, we proved that the matrix $\mathbf{R}_k$ is full rank and this means that the smallest singular value of $\mathbf{R}_k$ is strictly positive. However, while computing singular value decomposition, numerical approximations are performed and the singular values below a given threshold are assimilated to linearly dependent columns. In other words, we need to make a stronger statement and ensure that the smallest singular value stays above a numerical precision threshold in good probability.

To this end, we recall the result of [22, lemma 3.15]:

Lemma 3. *Suppose that k and ℓ are positive integers with $k \leq \ell$. Suppose further that G is a real $\ell \times k$ matrix whose entries are i.i.d. Gaussian random variables of zero mean and unit variance, and β is a positive real number, such that*

$$1 - \frac{1}{\sqrt{2\pi(\ell-k+1)}}\Big(\frac{e}{(\ell-k+1)\beta}\Big)^{\ell-k+1} \tag{2}$$

is nonnegative.

Then, the least (that is, the kth greatest) singular value of G is at least $\frac{1}{\sqrt{\ell}\beta}$ with probability not less than the amount in (2).

Since $\mathbf{R}_k$ in our case is a Gaussian random matrix of size $k \times k$, zero mean and variance $\frac{1}{k}$, then $s_{\min}$, the smallest singular value of $\mathbf{R}_k$, equals $\frac{\lambda_{\min}}{\sqrt{k}}$ with $\lambda_{\min}$ the smallest singular value of a matrix whose entries match the lemma above. Thus from this result, we can state that the cummulative density function of $s_{\min}$ is:

$$\mathbb{P}(s_{\min} < \frac{1}{\beta}) < \frac{e}{\beta\sqrt{2\pi}} \tag{3}$$

In practice, we need to ensure that the minimal singular value is above the predefined threshold of the rank estimate, that usually is around 10^{-13}. Knowing that the probability of $s_{\min}$ being below the numerical threshold is less than $\frac{e}{\sqrt{2\pi}}10^{-13} \approx 10^{-13}$, we can conclude that the claim we made theoretically for the rank still holds in practice with very high probability.

Now that we confirmed that $\mathbf{R}_k$ is full rank, even considering numerical approximations, we analyze the final projection $\mathbf{M} = \mathbf{U}_k\mathbf{R}_k$.

Lemma 4. *Let $\mathbf{M} = \mathbf{U}_k\mathbf{R}_k$ a matrix of size $N \times d$, with $\mathbf{U}_k$ and $\mathbf{R}_k$ as defined above. The two following statements are correct:*

$$\forall x \in \mathbb{R}^k, \exists y \in \mathbb{R}^d : \mathbf{U}_k x = \mathbf{M} y. \tag{4}$$

$$\forall y \in \mathbb{R}^d, \exists x \in \mathbb{R}^k : \mathbf{U}_k x = \mathbf{M} y. \tag{5}$$

That is $\mathbf{M}$ and $\mathbf{U}_k$ share the same column space.

Proof. Since $\mathbf{R}_k$ is full rank, its span is able to generate any matrix of $\mathbb{R}^{k\times d}$. Then, the projection of this full space onto $\mathbf{U}_k$ can form any matrix generated by the span of $\mathbf{U}_k$. □

Note that, although all the lemmas above assume $d \geq k$, we suggest using $d = k$ in practice since this is the minimal value for which the result holds and thus the one that will require the least computation.

Proof of theorem 1. From $\mathbf{M}$, we can find a set of k orthonormal vectors $\mathbf{B} = \{\mathbf{b_1}|\mathbf{b_2}|\dots|\mathbf{b_k}\}$, e.g., by applying an SVD. We obtain a decomposition such as $\mathbf{M} = \mathbf{B}\Sigma\mathbf{V}^\top$, with Σ a diagonal matrix and $\mathbf{V}$ an orthogonal matrix. This gives the following equality:

$$\mathbf{U}_k\mathbf{R}_k = \mathbf{M} = \mathbf{B}\Sigma\mathbf{V}^\top \tag{6}$$

and thus $\mathbf{U}_k$ and $\mathbf{B}$ have the same column space by definition. But since $\mathbf{B}$ and $\mathbf{U}_k$ also have the same shape and orthonormal columns, they necessarily relate to each other as $\mathbf{B} = \mathbf{U}_k\mathbf{Q}$, for some rotation matrix $\mathbf{Q} \in \mathbb{R}^{k\times k}$. □

Before moving on to the following of the paper, we would like to stress the fact that the theory described here does not use any assumption made on $\mathcal{L}$. Thus, the statements we make are also true for any matrix for which there exists a spectral decomposition. However, the sparsity of this matrix is key to a fast implementation using graph filtering as we will show next.

3.2 M as an approximation of $\mathbf{U}_k$

The matrix $\mathbf{B}$ has been shown to approximate $\mathbf{U}_k$ up to a rotation, which is perfectly fine for all common applications (e.g., embedding, spectral clustering, etc.). In the following lines, we wanted to present the quality of $\mathbf{M}$ as a direct approximation of $\mathbf{U}_k$. In the discussion below, we show that it could be enough in some situations to stop the procedure before the SVD step and reduce then the complexity of the algorithm.

Recall that $\mathbf{M}$ and $\mathbf{B}$ share the same column space (i.e., span$\{\mathbf{U}_k\}$) as we proved in Theorem 1 and have the same shape. The major difference between the two is that only the latter is composed of normalized columns. However, the distribution of the singular values of $\mathbf{M}$ is well known: it is the same as that of $\mathbf{R}_k$ since $\mathbf{U}_k$ has unitary columns. Moreover, the works of Marchenko and Pastur [23] contain lots of results regarding the study of Gaussian ensemble and Wishart matrices. They showed, among other things, that the eigenvalues of Wishart matrices follow a quarter circle law, which means that the distribution of any singular value of $\mathbf{M}$ is a normalized quarter circle of support $[0; 2]$ when $d = k$. On top of that, they proved that the expected value and the standard deviation of those eigenvalues tend to 1 as N becomes large. This means that in average, even with $d = k$, $\mathbf{M}$ is a very good candidate for the approximation of the subspace. The problem is that with the variance on the eigenvalue distribution, random samples hardly benefit from the expectation.

Meanwhile, the Johnson-Linderstrauss lemma says that with $d = \mathcal{O}(\log(N))$, the distances are almost preserved (up to a $(1 + \varepsilon)$ multiplicative factor) with high probability between

rows of $\mathbf{U}_k$ and rows of $\mathbf{M}$. Thus, it seems intuitive that picking more random signals would improve the repartition of the eigenvalues between 0 and 2 and concentrate around the mean. In fact, from the definition of the Marchenko-Pastur distribution, we have the following result:

Corollary 1 (Corollary 5.35 from [24])**.** *Let A be an $N \times n$ matrix whose entries are independent standard normal random variables. Then for every $t \geq 0$, with probability at least $1 - 2\exp(-t^2/2)$ one has $\sqrt{N} - \sqrt{n} - t \leq s_{\min}(A) \leq s_{\max}(A) \leq \sqrt{N} + \sqrt{n} + t$.*

In our case, the entries are Gaussians of variance $\frac{1}{N}$ and the result becomes:

$$1 - \sqrt{\frac{k}{d}} - \frac{t}{\sqrt{d}} \leq s_{\min}(\mathbf{R}_k) \leq s_{\max}(\mathbf{R}_k) \leq 1 + \sqrt{\frac{k}{d}} + \frac{t}{\sqrt{d}} \tag{7}$$

We conclude that the more the matrix $\mathbf{R}_k$ is flat (i.e., $d > k$), the more its eigenvalues are concentrated around 1 in good probability, which confirms our intuition.

4 Computational aspects of subspace approximation

Now that our main theoretical result is established, we look into its practical implementation, while focusing on efficient solutions. First, we present a solution on how to find the cutoff eigenvalue λ_k. Then, we show our choice for the actual filter design using polynomials enabling fast filtering operations while limiting the problems caused by the approximation. Finally, we describe our algorithms and analyze their complexity.

4.1 Estimation of λ_k

The computation of $\mathbf{M}$ described above depends on the quality of the filter g and the determination of its cutoff frequency λ_k which is not known a priori. A standard method is to use eigencount techniques such as the one proposed in [25]. In this work, the authors used the fact that the energy retained by an ideal low-pass filtering of random signals with cutoff frequency λ_k, called g_{λ_k}, is proportional to the number of eigenvalues that are smaller than λ_k. Mathematically, we have:

$$\mathbb{E}[\|g_{\lambda_k}(\mathcal{L})\mathbf{R}\|_F^2] = |\{\lambda : \lambda \leq \lambda_k\}|. \tag{8}$$

Thus, by dichotomy, one could approximate the desired threshold value λ_k for our filter since we want it to capture exactly k eigenvalues and we know that $\lambda_{\max} \leq 2$ for normalized Laplacians. Unfortunately, each step of the dichotomy requires $\mathcal{O}(k)$ filterings and the dichotomy must be applied $\mathcal{O}(\log(N))$ times, without making strong assumptions on the distribution of the eigenvalues over the spectrum. Thus, the estimation of λ_k such as defined and used in [17] is $\mathcal{O}(m|\mathcal{E}|k\log(N))$, which is above the complexity of all the rest of our problem.

We propose now an accelerated version of the eigencount technique for the determination of the threshold of the filter that will not increase the complexity of the overall algorithm. We first assume that the eigenvalues are distributed evenly over the spectrum (between 0 and $\lambda_{\max}$). Thus, on average, the k^{th} eigenvalue should be $\mathbb{E}(\lambda_k) = \frac{k}{N}\lambda_{\max}$. However, one will not find the exact count systematically on the first guess, due to the randomness of the process and the non-uniformity of the eigenvalue distribution in practice. We suggest thus to iterate with the assumption of local uniformity of the distribution of the eigenvalues until the goal is reached. In practice, this means that after picking $\lambda(0) = \frac{k}{N}\lambda_{\max}$, one should apply the eigencount technique to compute the approximation of the real number of eigenvalues below $\lambda(t)$ in the graph of study, called $n_\lambda(t)$, and iterate with $\lambda(t+1) = \frac{k}{n_\lambda(t)}\lambda(t)$ until the targeted count is achieved with good precision (see Algorithm 2 for details). As the

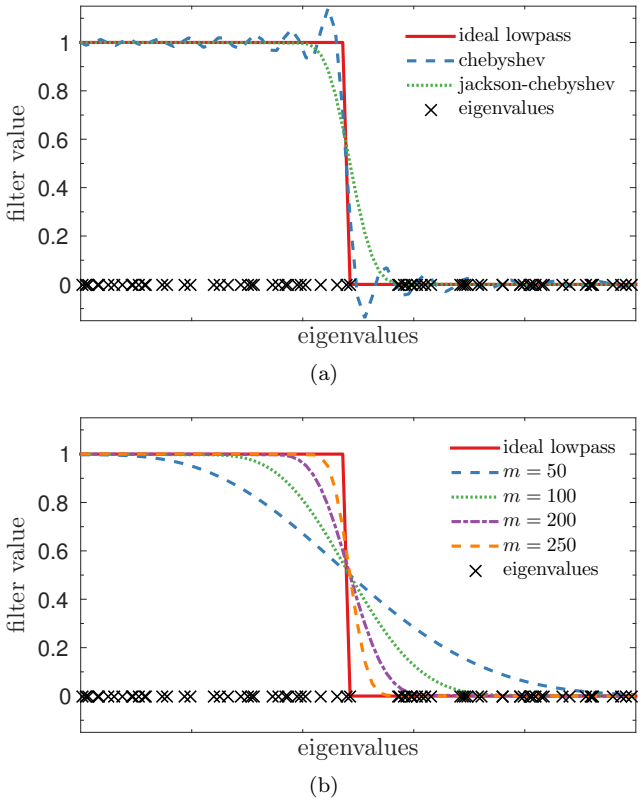


Figure 1: The effect of approximating a step function with polynomials. The solid red line is the ideal step function. The black crosses represent the eigenvalues. The approximation using Jackson-Chebyshev polynomials (dotted line) is compared with Chebyshev polynomial approximation (dashed line) of same order m in (a). Jackson-Chebyshev approximations with different orders m are compared in (b).

number of iterations does not depend on N but only of the local eigenvalue distribution, a good precision can be achieved with a constant number of iterations. The cost in number of operation of this accelerated version is thus $\mathcal{O}(m|\mathcal{E}|k)$ which is acceptable since it is of the same order than the remaining of our method.

4.2 Acceleration using fast filtering

The construction of the matrix $\mathbf{M}$ in the previous section requires the knowledge of the first k eigenvectors of the graph Laplacian. This knowledge is very costly for large graphs (N large) since it requires a partial SVD of a $N \times N$ matrix, which we try to avoid in the first place. Fortunately, as we explained before, the product $\mathbf{U}_k\mathbf{U}_k^\top$ corresponds to a graph filtering with $g(\mathcal{L})$, g being an ideal low-pass filter:

$$g(\lambda) = \begin{cases} 1 & \lambda \leq \lambda_k \\ 0 & \lambda > \lambda_k \end{cases}$$

Since we cannot afford the cost of exact filtering, we use a polynomial approximation of the filter $g(\mathcal{L})$. There exist several methods using powers of the Laplacian that allow approximating such filters with polynomials (Chebyshev [20], Jackson-Chebyshev [25] or Lanczos [21] polynomials). In the task at hand, the Jackson-Chebyshev polynomial approximation is the best suited to approximate the step function of $g(\mathcal{L})$ since it avoids the Gibbs effect of Chebyshev polynomials as can be seen in Fig. 1a.

The quality of the approximation is based on the order of the polynomial, directly related to

the number of coefficients to compute. If we define m as the highest degree of the polynomial, we can show that the error of approximation decreases as m increases. This effect is shown in Fig. 1b where we can see the convergence to the ideal low-pass with an increasing value of m. But since the complexity of the filtering increases linearly with m one cannot let it become too large. In particular, we cannot let m be $\mathcal{O}(N)$ since it would have a huge impact on the overall complexity.

Let us remind here that the filter approximation needs to be correct only on the discrete values given by the eigenvalues. Indeed, the approximation does not need to fit closely g while the discrete values that the filter takes on the eigenvalues are correct. In our case, since we only want to approximate a step function, we need the value of the filter to be equal to 1 for $\lambda_0, \lambda_1, \ldots, \lambda_{k-1}$ and 0 for $\lambda_k, \lambda_{k+1}, \ldots, \lambda_{N-1}$. Two situations could lead to the non-respect of this condition. The estimated cutoff eigenvalue can be wrong or the order of the polynomial can be too small.

If the order m is too small, then, as can be seen in Fig. 1b for $m = 100$, the filter will be below 1 for a few eigenvalues below λ_{k-1}, and above 0 for a few eigenvalues after λ_k. If the estimated cutoff eigenvalue is a bit off, a similar situation will happen, with a shift towards lower or higher frequencies. In both cases, the value of the filter will still be 1 up to some eigenvalue λ_j, then monotonically decreasing to 0 up to some eigenvalue λ_l and 0 up to λ_{N-1}, with $\lambda_j < \lambda_k < \lambda_l$. In such a case, the filter will have non-zero coefficients in the range $[\lambda_k, \lambda_l]$ and thus, $\mathbf{M}$ will be contaminated by some elements of the space $\mathbf{U}_{[k+1,l+1]}$. However, these contributions will not appear too much in the energy of $\mathbf{M}$ since the coefficients of the filter for the eigenvalues bigger than λ_k are smaller than all coefficients for the range $[\lambda_0, \lambda_{k-1}]$. Since our final approximation $\mathbf{B}_k$ is done using an SVD of $\mathbf{M}$, then $\mathbf{B}_k$ will be the best rank k approximation of $\mathbf{M}$ by minimizing the energy of the residuals. Overall, as one can verify in the experiments of section 5, $\mathbf{B}_k$ will provide features that remain very good for the various applications that we develop, even with a very low polynomial order.

4.3 Algorithms

We propose in this section to summarize the procedure to obtain the approximation of the subspace $\mathbf{U}_k$ based, on one side, on the theoretical development of section 3.1, and, on the other side, on the practical considerations of sections 4.1 and 4.2.

Algorithm 1 summarizes the steps of our method to approximate the Laplacian eigenspace $\mathbf{U}_k$ from data points. If a graph is not provided with the data, a k-NN graph can be constructed and its associated Laplacian computed beforehand. The algorithm takes a graph and a number k as input and outputs a set of k approximated eigenvectors of the graph Laplacian.

Algorithm 1 Eigenspace Approximation

1: Generate $\mathbf{R}$ with $d = k$ cf. Section 3.1
2: Estimate λ_k cf. Algorithm 2
3: Compute the approximated graph filter g cf. Section 4.2
4: Apply filtering: $\mathbf{M} = g(\mathcal{L})\mathbf{R}$
5: Compute an economic SVD: $\mathbf{USV} = \text{SVD}(\mathbf{M})$
6: Return the left singular vectors $\mathbf{U}$

Algorithm 2 presents in details the strategy described in section 4.1 for the accelerated estimation of λ_k. The main assumption here is that the distribution of the eigenvalues is uniform by part over the spectrum. We thus try to reach such segment of the spectrum where uniformity applies to fasten the discovery of the value of λ_k. Since some parts of the spectrum can be empty due to eigengaps for some classes of graphs, we implemented a dichotomic step to get a broad spectrum distribution estimate if the search does not progress.

Algorithm 2 Estimation of λ_k

Input: k, λ_{max} and $\mathcal{L}$
Output: λ_k (the k^{th} eigenvalue of $\mathcal{L}$)

Initialize: $\lambda_{lb}, c_{lb}, iter, c_{est} \leftarrow 0$
$\lambda_{ub} \leftarrow \lambda_{\max}, c_{ub} \leftarrow N$
$\lambda_{est} \leftarrow k\frac{\lambda_{\max}}{N}$
Generate $\mathbf{R}$ with $d = k$
while $c_{est} \neq k$ and $iter < max_{iter}$ **do**
    Compute approximated graph filter g with $\lambda = \lambda_{est}$
    $c_{est} \leftarrow \|g(\mathcal{L})\mathbf{R}\|_F^2$
    if $c_{est} < k$ **then**
        $\lambda_{lb} \leftarrow \lambda_{est}$
    else
        $\lambda_{ub} \leftarrow \lambda_{est}$
    end if
    if $c_{lb} = c_{est}$ or $c_{ub} = c_{est}$ **then**
        $\lambda_{est} \leftarrow \frac{\lambda_{lb}+\lambda_{ub}}{2}$
    else
        if $c_{est} < k$ **then**
            $c_{lb} \leftarrow c_{est}$
        else
            $c_{ub} \leftarrow c_{est}$
        end if
        $\lambda_{est} \leftarrow \lambda_{lb} + (k - c_{lb})\frac{\lambda_{ub}-\lambda_{lb}}{c_{ub}-c_{lb}}$
    end if
end while
return λ_{est}

4.4 Complexity analysis

Steps 1 and 3 of Algorithm 1 are nonsignificant in the analysis of the overall complexity. We focus here on steps 2, 4 and 5 for which the number of operations is studied in details. Using fast filtering operations, applying our method consists of k graph filtering operations at step 4, which is $\mathcal{O}(m|\mathcal{E}|k)$, with m the order of the polynomial approximation of the filter. The SVD performed in step 5 has an additional cost of $\mathcal{O}(k^3)$ for a tall matrix of size N by k like here. Finally, step 2 takes $\mathcal{O}(m|\mathcal{E}|k)$ if we consider the amelioration proposed in section 4.1. Thus, the overall complexity of our method is $\mathcal{O}(m|\mathcal{E}|k + k^3)$.

Comparison with IRLM [10] As reminded above, the complexity of IRLM is $\mathcal{O}(h(|\mathcal{E}|k+ k^2N + k^3))$ with h a convergence factor. Thus, assuming h and m have similar orders, the IRLM needs at least $\mathcal{O}((h-1)k^2N)$ more operations than our method. In any reasonable application, we will have either $k < N$ or $k \ll N$, thus, the term $\mathcal{O}(hk^2N)$ will be larger than the term $\mathcal{O}(hk^3)$.

Comparison with CSC [17] Although the method presented in CSC is not directly an eigenspace estimation method, it does use the same mechanics of filtered random signals on the graph to obtain the spectral features. The number of filtering needed is d, which has to be larger than a threshold given by results presented in Theorems 3.2 and 3.4 of their paper. To simplify, we can say that $d = \gamma \log(\alpha k \log(k))$ where γ and α are influenced by the precision of the distance preservation and the probability that the distance is preserved. Note that even with medium precision (e.g., $\mathcal{O}(10^{-1})$), the constants γ and α will be large (i.e., $\mathcal{O}(10^3)$). This means that the overall complexity for the spectral features estimation will cost $\mathcal{O}(m|\mathcal{E}|\gamma \log(\alpha k \log(k)))$ operations. Finally, the $\mathcal{O}(\log(N))$ filterings required to estimate λ_k have an added cost of $\mathcal{O}(m|\mathcal{E}| \log(N))$.

If we compare the complexity of our proposed method with the CSC we get the difference of number of operations:

$$\begin{aligned}\Delta &= m|\mathcal{E}|k + k^3 - m|\mathcal{E}|(d + \log(N)) \\ &= m|\mathcal{E}|(k - d - \log(N)) + k^3 \\ &= m|\mathcal{E}|(k - \gamma \log(\alpha k \log(k)) - \log(N)) + k^3\end{aligned}$$

For sparse graphs we can assume $|\mathcal{E}| = c_d N$, with c_d the average node degree, which gives:

$$\Delta = m c_d N(k - \gamma \log(\alpha k \log(k)) - \log(N)) + k^3.$$

In order to finish the comparison, we now need to make hypotheses on the relation between k and N.

If we assume that $k = \mathcal{O}(log(N))$, then, for N large

$$\begin{aligned}\Delta &= m c_d N(\log(N) - \gamma \log(\alpha k \log(k)) - \log(N)) + \log^3(N) \\ &= \log^3(N) - m c_d N \gamma \log(\alpha k \log(k)) < 0,\end{aligned}$$

with the last step following from the fact that $\log(\alpha k \log(k)) > 1$ and $\mathcal{O}(log^3(N)) < \mathcal{O}(N)$. This means, that for this regime, our method is cheaper than CSC, for large N.

If we assume that $k = \mathcal{O}(\sqrt{N})$, then, for N large

$$\begin{aligned}\Delta &= m c_d N(\sqrt{N} - \gamma \log(\alpha N^{\frac{1}{2}} \log(N^{\frac{1}{2}})) - \log(N)) + \sqrt{N^3} \\ &= N(\sqrt{N}(m c_d + 1) - \gamma \log(\frac{\alpha \log(N)}{2}) - \frac{\gamma + 2}{2} \log(N)) \\ &> 0,\end{aligned}$$

with the last step coming from the fact that $\gamma > 1$ and $\mathcal{O}(\sqrt{N}) > \mathcal{O}(\log(N))$. This means that for this regime CSC will be cheaper than our method for large enough N.

From the two cases described above we can assess that if $\mathcal{O}(1) \leq k \leq \mathcal{O}(\log(N))$ our method is cheaper and if $\mathcal{O}(\sqrt{N}) \leq k \leq \mathcal{O}(N)$ then CSC is cheaper. Note that in both cases the order of the filter m was kept constant, but that both results hold for any m, even with $m = \mathcal{O}(N)$.

5 Experiments

In this section, we provide experiments whose objective is to show how our proposed methods behave in practice. First, we want to ensure that our proposed algorithms do fulfill their goals, i.e., that they provide accurate enough results and do so efficiently. Second, both as illustrations and practical applications, we show the performance of our eigenspace approximation method on typical clustering and visualization tasks.

The experiments were performed with the GSPBox [26], an open-source software. As we follow reproducible research principles, our implementations and the code to reproduce all our results is open and freely available[1]. Since our methods use random signals, it is expected that the results shall be slightly different in the details, but overall consistent.

[1] Available at `https://lts2.epfl.ch/reproducible-research/fears/`

5.1 Time performance analysis

Since the complexity analysis in Section 4.4 only covers asymptotically large N, it is also interesting to look at the cost of the algorithms for actual implementations and realistic values of N and k. In addition to the eigenspace estimation with IRLM (eigs) and the k-dimensional spectral features of Compressive Spectral Clustering (CSC) mentioned in Section4.4, we consider the power method described in [16] (power).

The data on which the different methods are evaluated consists of N points of small intrinsic dimension which are randomly drawn. In addition, a knn graph with 10 neighbors is constructed from the data points. Each method is run with fixed parameters and the time is measured in total CPU time to completion. The results of the experiments can be seen in Fig. 2.

Fig. 2a shows the time needed in function of k with N fixed and for small values of k. The first note is that the power method does not scale well with k and is exceedingly time-consuming for everything other than very small values of k for which it performs well. Since it is order of magnitudes slower for the parameters used in the other experiments, it is not displayed in the remaining figures to keep readability. Fig. 2b is the same as Fig. 2a for larger values of k. We see, as expected in accordance with the complexity analysis, that above a threshold corresponding to $\sqrt{N}$ (i.e., 100), our method performs better than eigs and worse than CSC.

Fig. 2c shows the results for an exponentially growing N and $k = \log(N)$. In this regime, our method outperforms both eigs and CSC for all values. The regime $k = \sqrt{N}$ is presented in Fig. 2d where we can see that our method performs best up to $N = 10^6$. Above this value, CSC is best. Note that results above $N = 10^6$ for this regime are not shown due to memory limitations for eigs.

Combined, those results confirm the conclusions drawn from the complexity analysis of Section 4.4. First, except for very small values of k, eigs is the most time-consuming method, even though it benefits from very optimized implementations. Second, for the $\log(N)$ regime, our method performs best for all values of N. For the $\sqrt{N}$ regime, our method is cheaper than CSC for $N < 10^6$. Above the limit $k = \sqrt{N}$, CSC is the cheapest method. As a final remark on these results, we need to point out that, contrarily to the other methods considered in this experiment, CSC does not compute an eigensubspace per se but only k-dimensional features allowing good pairwise distance measurements between data points.

As a last remark on timing, we want to call attention to the fact that when filtering multiple random signals, all filtering operations are independent. Indeed, the signals are independent by definition and both the polynomial coefficients of the filter and the Laplacian are unaltered by the successive filtering operations. The filtering operations in our algorithms could thus easily benefits from a parallel implementation.

5.2 Quality of approximation for various graphs

In this section, we measure the accuracy of our algorithms for different classes of graphs and for different values of k and N. In particular, we wish to evaluate two things: on one hand, the quality of approximation of the eigenspace $\mathbf{U}_k$ with Algorithm 1 and on the other hand the precision and efficiency of our accelerated eigencount method with Algorithm 2.

The graphs chosen for this experiment are well-known classes in the field and have various spectral properties. Here is a list of all graphs with short descriptions:

- **Sensor network**: A graph of a synthetic sensor network, which represents randomly positioned sensors connected in a knn fashion.
- **SBM**: Stochastic Block Model graphs model social networks or community graphs

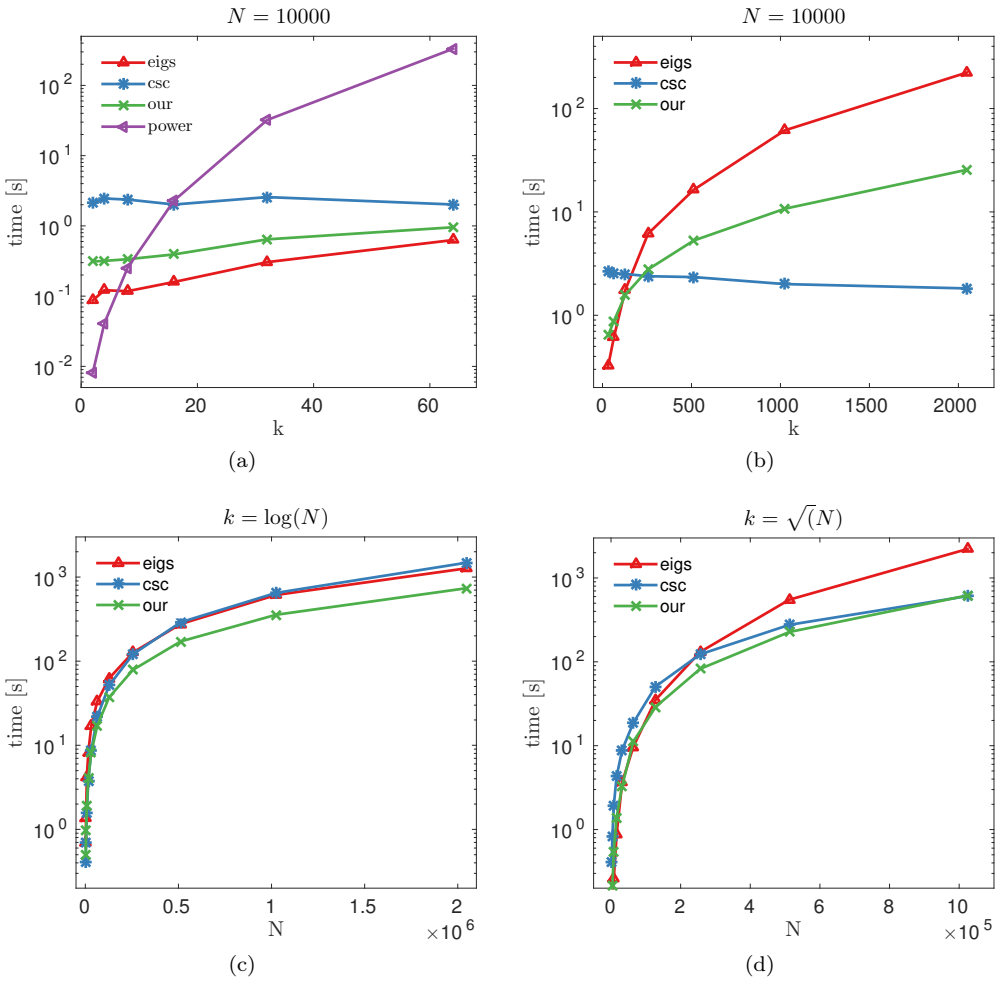


Figure 2: Comparison of CPU time needed between different methods for the estimation of an eigensubspace of dimension k. In (a) and (b) N is fixed and k increases. In (c) and (d) k varies in function of N in two regimes ($k = \log(N)$ and $k = \sqrt{N}$ respectively). Time axis are in log-scale.

and are known to be clusterable (and thus possesses eigengaps).

- **Swissroll**: This graph is a knn graph of the famous Swissroll manifold, a point cloud drawn from a rolled 2D surface in 3D.
- **Bunny**: This graph is the knn graph constructed from the 3D point cloud of the Stanford bunny.
- **Image graph**: This graph is created by connecting the pixels of an image using similarity of patches. The image of interest is the grayscale image of Barbara, a natural image often used in image processing.
- **Road network**: This graph represents the Minnesota road network (originally from the MatlabBGL library).

In order to measure the quality of the approximated eigenspace (up to a rotation), we introduce a measure of the amount of energy which is preserved when the approximated eigenspace is projected on the real eigenspace computed with exact methods. If we note the approximated eigenspace as $\mathbf{B}_k$ and the exact eigenspace $\mathbf{U}_k$, the normalized energy kept by the projection is:

		Sensor network	SBM	Swiss-roll	Bunny	Image	Road network	
		N = 10 000	N = 10 000	N = 10 000	N = 2503	N = 16 384	N = 2 642	
ME	exact	0.86 ± 0.01	1.00 ± 0.01	0.86 ± 0.02	0.99 ± 0.01	0.91 ± 0.01	0.93 ± 0.01	0.92
	standard	0.80 ± 0.03	0.95 ± 0.05	0.79 ± 0.03	0.94 ± 0.05	0.86 ± 0.04	0.90 ± 0.05	0.87
	fast	0.80 ± 0.03	0.96 ± 0.04	0.79 ± 0.03	0.95 ± 0.04	0.86 ± 0.04	0.90 ± 0.04	0.88
IT	standard	14.62 ± 0.90	**5.32** ± 1.58	4.68 ± 0.62	8.74 ± 1.77	13.06 ± 1.58	11.34 ± 1.22	11.29
	fast	**3.02** ± 0.71	9.36 ± 1.06	**2.86** ± 0.70	**4.48** ± 1.25	**3.12** ± 0.75	**3.06** ± 0.51	**4.31**
KD	standard	0.60 ± 0.53	2.46 ± 4.92	0.52 ± 0.58	0.36 ± 0.53	0.34 ± 0.48	0.36 ± 0.48	0.79
	fast	**0.00** ± 0.00	**1.00** ± 1.01	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.17**

Table 1: Quality measure of our proposed methods for eigenspace estimation and λ_k estimation*. For all experiments, the following parameters were used: the order of the polynomial approximation $m = 500$, $k = 25$, $\epsilon = 10^{-1}$ for the standard eigencount method and the maximum number of iterations in fast is 10. Bold face numbers are the best score between two lines. The last column is the average over all graphs. average mean energy (ME) (as in eq. 9). The ME measure is between 0 and 1 and higher values are better. The average number of iterations is noted IT ; smaller values are better. The mean squared deviation from k is noted KD ; smaller values are better. In ME, exact denotes the score computed using the true λ_k. For everything else, λ_k is estimated either with the dichotomy method proposed in [17] (standard) or with our proposed method as in Algorithm 2(fast).

$$E(\mathbf{B}_k, \mathbf{U}_k) = \frac{1}{k}\|\mathbf{B}_k^T \mathbf{U}_k\|_F^2. \tag{9}$$

We chose to use the normalized energy to score the quality of the estimated eigenspace as it gives a number between 0 and 1 where higher values mean better approximation.

In order to compare our accelerated eigencount method with the reference dichotomy implementation of [17] (abbreviated fast and standard respectively in the table), we used two measures. First, the number of iterations required until convergence, which is adequate since the workload per iteration is the same in the two algorithms. Finally, we measure how close to the actual k have the algorithm converged as the mean squared deviation from k. This last measure is useful to state if the method was able to converge with respect to the current random matrix used for estimation, not with respect to the actual value of λ_k.

The results of all measures for the various graphs described above are reported in Table 1. Due to the randomness of the methods we evaluate, all experiments are averaged over 50 realizations and the standard deviation is indicated for all measures.

If we first focus on the upper part of Table 1 we can see that the measure of the energy (ME) using the true cutoff λ_k shows an average above 90% of precision over all graphs with a perfect score for very clusterable graphs (such as SBM) and lower values for more difficult graphs (such as Sensor network). The trend is similar using estimated values for λ_k both with the standard and fast methods. Using the approximated cutoffs lowers the score of about 5%. Using the fast method leads to marginally better results. One very interesting fact regarding these results is that both the λ_k estimation step and the eigenspace approximation contribute to the lost energy in approximately equal amounts. This tends to indicate that it is important to balance the computational effort between the two steps and not favoring one against the other.

On the middle part of Table 1, we can see the first measure reported for the eigencount evaluation. The number of iterations needed to compute λ_k (IT) is lower with the fast method for all but the SBM graph. On average, the fast method is 2.5 times faster than the standard method. For the SBM graph, fast is close to its maximum number of iterations meaning that the eigencount hardly converged. This result can be easily explained by the fact that the eigenvalue distribution for SBM is known to be highly non-uniform, especially for low frequencies, which is partly incompatible with the local uniformity hypothesis assumed by the fast method.

On the lower part of Table 1 the precision of the estimated k (KD) is reported. Both the fast and standard method converge most of the time, with a better overall convergence of the former which converges exactly to the true value except for SBM. This could be expected from the high number of iterations needed for this specific graph.

From those results, we can see that the quality of the estimated subspace computed using our proposed method is decent, while not perfect. The imprecision coming both from the approximation in the filter design and cutoff eigenvalue estimation. Our scheme for accelerated λ_k estimation is faster than the reference method and provide very good results.

5.3 Clustering

This experiment proves the capability of our filtered signals (us) to produce an assignment for the data points. We will compare the results obtained by our method to Spectral Clustering (SC) [2] and Compressive Spectral Clustering (CSC) [17]. We will also see that the compressive step of the latter can be used with k filtered signals instead of d.

Spectral clustering Spectral clustering is a very famous method that follows directly from the relaxation of NCut for k classes. It states that the k eigenvectors associated with the smallest eigenvalues of $\mathcal{L}$ are the optimal solution of the optimization problem of NCut. Thus, by computing the eigensubspace $\mathbf{U}_k$ one easily gets a very good assignment for the data partitioning problem since the k-means solution over the rows of the matrix $\mathbf{U}_k$ gives a standard discrete partition of the data points. However, computing spectral clustering on large graphs is not to be considered due to the runtime complexity of the method ($\mathcal{O}(N^3)$ for exact methods, $\mathcal{O}(k^2N)$ with IRLM).

Compressive spectral clustering In this work, the authors replaced the features formed by the eigenvectors with filterings of random signals on the graph $\mathcal{G}$. They propose a minimal number of signals to filter in order to preserve the distances between any two points in the data set. There only remains to apply k-means on the filtered signal to obtain an assignment identical to spectral clustering. Their second contribution is to show that k-means can be compressed, in the sense that only a subset of the nodes needs to be assigned with this costly method. The remaining labels can be inferred by solving an optimization problem using graph regularization.

5.3.1 Synthetic case: Stochastic Block Model

For this experiment, we use a Stochastic Block Model (SBM) with $N = 5\,000$ nodes and $k = 20$ clusters. We set the average degree of the nodes to $s = 16$ and the nodes are associated at random with a particular class (the ground truth for the assignment). Then, an edge between two nodes exists with probability p if the two nodes belong to the same class (intra-cluster probability) and with probability $q < p$ if they belong to different clusters (inter-cluster probability). We generate several graphs with different ratios $\varepsilon = \frac{q}{p}$ (the larger ε, the harder the community detection) to evaluate our clustering capabilities in the task.

The evaluation of the presented methods is performed using the adjusted Rand similarity index [27] between the SBM ground truth and the resulting assignments. All results presented here are averaged over 50 realizations in each setup. By looking at Fig. 3 we can first observe that our method is the one that approximates the best the results of SC. It is not necessarily the method achieving the best rand index as ε increases but the ground truth is set before the edges are created. Thus, for relatively large values of ε, it might not make sense to keep this assignment for clustering purposes. In our view, spectral clustering is the target to fit at best. Moreover, notice that the order of the polynomial approximations alters

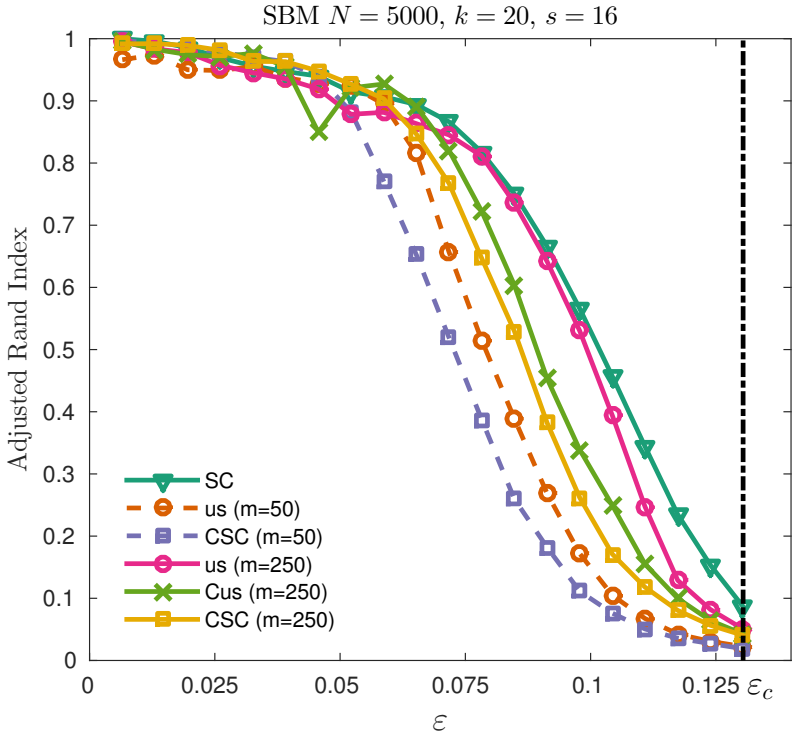


Figure 3: Study of the clusterability of Stochastic Block Models for various values of ε, representing how well the graph can be split into clusters. Our method is the best to approximate the result of spectral clustering.

the result of the clustering in both our method and CSC. Finally, Cus represents the result of our features assigned with the compressive step of CSC instead of the full k-means. We see that k-means is more faithful to spectral clustering than the regularized label diffusion on the graph.

5.3.2 Real-world example: Amazon co-purchasing network

In addition to the synthetics SBM graphs, we want to go further and show that this also works well for real-world data sets. To this end, we consider the problem of clustering the Amazon co-purchasing network [28] that has also been evaluated for the study of CSC. The graph is composed of 334 863 nodes and 925 872 edges[2]. No clear ground truth can be used to compare against since the given information is the belonging of products to categories with overlaps. We decided to reproduce the experiment published in [17], adding our method to the benchmark. We measured the resulting assignments with two measures: the modularity score [29], used to determine whether a given partition is separating the network efficiently, and the adjusted Rand similarity index compared to the result of SC, used to identify the resemblance of the two assignments.

In Table 2 we first show the performance of the different algorithms with 3 different numbers of clusters: 250, 500 and 1 000. We split the timing into two parts, one for the feature extraction process and the other for the assignment based on these features. We see that consistently the features extracted using random signal filtering are faster to compute than those requiring partial eigendecomposition. We also notice that until $k = 500$, k-means is an efficient method for the assignment of the points to the clusters, it is even 5 times faster than the compressive assignment for $k = 250$ in our experiment. However, when k becomes larger, using the compressive method of CSC (also applied in Cus) is helping greatly to

[2]Available at `http://snap.stanford.edu/data/com-Amazon.html`

	$k = 250$	$k = 500$	$k = 1\,000$
SC	14.37min + 2.13h	25.09min + 14.96h	55.63min + 106.87h
us	0.12min + 2.55h	0.19min + 22.75h	0.52min + 104.82h
Cus	0.12min + 11.36h	0.19min + 17.22h	0.52min + 58.46h
CSC	2.34min + 9.74h	3.73min + 21.07h	2.61min + 35.47h

Table 2: Timing of clustering for Amazon data set. All values represent one experiment and the order of the polynomial approximation is $m = 500$. Each experiment is split into two steps: the computation of the features (in minutes), and the assignment from the features to a cluster (in hours).

	SC	us		Cus		CSC	
	mod	mod	rand	mod	rand	mod	rand
$k = 250$	0.344	0.387	0.884	0.588	0.711	0.764	0.509
$k = 500$	0.507	0.605	0.818	0.759	0.677	0.818	0.586
$k = 1\,000$	0.663	0.638	0.851	0.815	0.780	0.798	0.749

Table 3: Evaluation of clustering for Amazon data set. All values are representing one experiment and the order of the polynomial approximation is $m = 500$. The modularity score ([29]) is noted mod and the adjusted Rand similarity index ([27]) rand.

reduce the overall time of the computation, earning a factor 2 speedup between us and Cus.

Next, we consider the efficiency of the clustering reported in Table 3, where two important observations stand out. On one hand, the best modularity is achieved using CSC and we see that our method, with the use of the compressive step, tends to similar results with increasing k. On the other hand, the adjusted Rand similarity index clearly shows that our method is assigning the nodes very similarly to SC. This is an expected behavior since the goal of our method is to reconstruct the set of the k first eigenvectors used as features in SC.

5.4 Visualization

In this last experiment, we show how our method can be used in the context of visualizing high-dimensional data, since eigenspaces are commonly used for dimensionality reduction in this context. We wish to see how our proposed method behaves first in a very simple synthetic example and second for real-world data sets of larger size. For this task we compare the following visualization algorithms:

Laplacian eigenmaps Belkin and Niyogi [3] proposed to solve the generalized eigenvalues problem $\mathcal{L}\mathbf{y} = \lambda D\mathbf{y}$ where y is called the Laplacian eigenmaps. This method is interesting to validate the fact that our method finds a good approximation of $\mathbf{U}_k$ because it finds the eigenspace of the random walk Laplacian. Indeed, if we define the random walk Laplacian as $P = D^{-1}\mathcal{L}$ then the equation above can be rewritten as $P\mathbf{y} = \lambda\mathbf{y}$. Thus, Laplacian eigenmaps aims at finding the eigenspace of P and use it as an embedding for visualization. We implemented the method in Matlab with the eigs eigensolver which uses the IRLM algorithm.

t-SNE [30] a famous state-of-the-art technique for visualization which enhanced the Stochastic Neighbor Embedding method [5]. The use of a heavy-tail distribution for the embedded points probabilistic model allows avoiding the crowding effect and at the same time gives rise to an easier optimization problem. The original implementation having an $\mathcal{O}(N^2)$ complexity, the Barnes-Hut accelerated version is often used for large data sets since

it has a $\mathcal{O}(N \log(N))$ complexity. We used the C++ implementation of the Barnes-Hut t-SNE for our experiments[3].

LargeVis [31] a recent technique based on graph visualization which aims at solving the scalability problems of state-of-the-art methods such as t-SNE. Its first contribution is to accelerate the graph construction step by using an approximated k-NN graph construction method. Second, it formulates the embedding problem as a probabilistic model which keeps similar vertices close to each other and dissimilar vertices apart. Inspired by negative sampling techniques they propose to optimize the probabilistic model using independent stochastic gradient descent steps. The C++ implementation of the algorithm was used for the experiments[4].

5.4.1 Toy example: the Swissroll

In this first small experiment, we wish to assess the validity of using our proposed method of eigenspace estimation for visualization on a simple toy example. We will compare the results obtained by our method only with Laplacian Eigenmaps as we would like to verify that we get similar results.

For this experiment, we use a classical Swissroll graph for which we compute a 2 dimensional embedding. The Swissroll is computed by sampling its continuous manifold in the following way: given a set of randomly drawn angles θ in $[a\pi, b\pi]$ the coordinates are set as $x = \theta\cos(\theta)$, y drawn uniformly in $[0, 1]$ and $z = \theta\sin(\theta)$. A knn graph with 10 neighbors is constructed from the data points. For this experiment, the normalized Laplacian was used for all methods.

The resulting embeddings are shown in Fig. 4. The colormap is a linear function of θ. The first thing to notice is that all embeddings are very smooth with respect to θ. The second interesting fact is that $\mathbf{B}_k$ indeed seems to be a good approximation of the Laplacian eigenmaps up to a rotation as they have very similar shapes. This tends to validate that the method indeed provides a good approximation of $\mathbf{U}_k$. In addition, in this specific example, while embedding with M gives a smooth result, the normalization step provided by the SVD is necessary to get a good enough visualization. This observation makes sense as for visualization very few random signals are used to get M, which, as discussed in Section 3.2, is not sufficient to have an excpectation effect smoothing the variance on the eigenvalues. This scaling is normalized by the final SVD step, which is not costly for visualization tasks since k is very small.

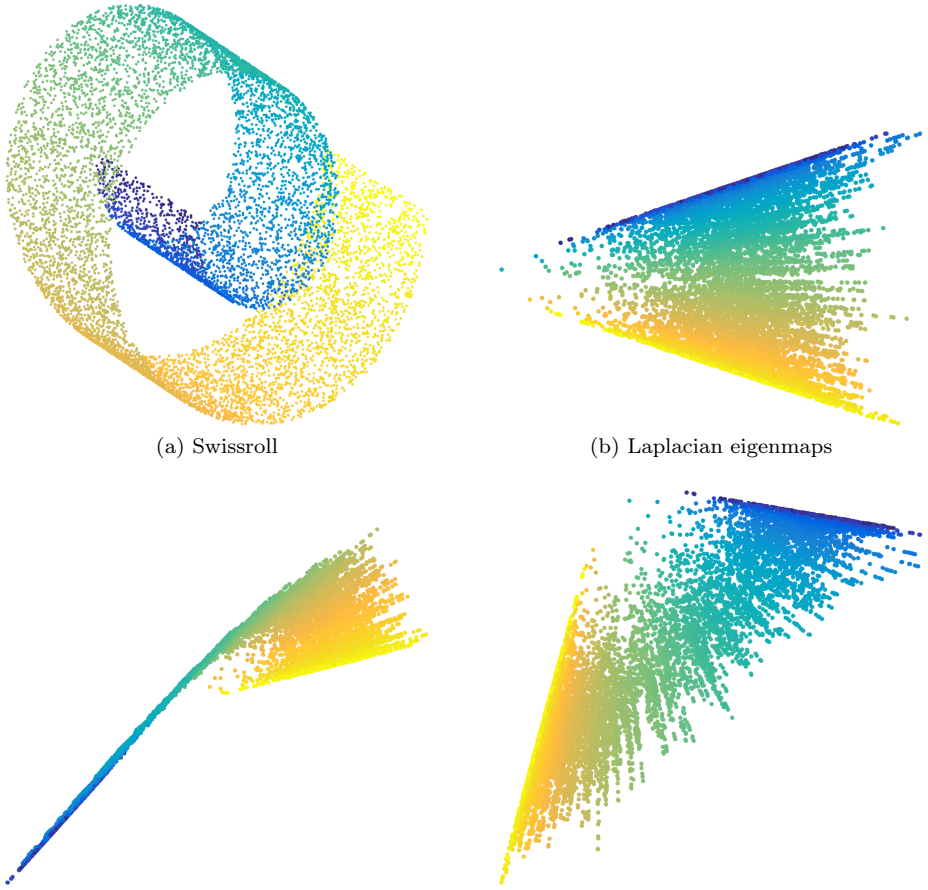
5.4.2 Real-world data sets

In this second experiment, we will consider large scale real-world examples and compare our method with existing approaches. We will use the two following data sets:

MNIST a well known data set of handwritten digit images, from which we take all 70 000 data points [5].

LiveJournal a data set from the LiveJournal social network. The graph used is the largest connected componant of the complete graph wich has 3 997 962 nodes[6].

[3] Available at `https://github.com/ninjin/barnes-hut-sne`
[4] Available at `https://github.com/lferry007/LargeVis`
[5] Available at `http://yann.lecun.com/exdb/mnist/`
[6] Available at `http://snap.stanford.edu/data/com-LiveJournal.html`



(a) Swissroll (b) Laplacian eigenmaps

(c) Our method with $\mathbf{M}$ (d) Our method with $\mathbf{B}_k$

Figure 4: The Swissroll point cloud (a) with 10 000 nodes and its 2D embeddings using Laplacian eigenmaps (b), our proposed fast eigenspace estimation method prior to the SVD step (c), and after the SVD step (d).

In Fig. 5 we can see the visualizations of the MNIST data set, where the colormap comes from the labels. The first observation is that both Laplacian eigenmaps and our proposed method yield similar results. Both do not achieve a very good separation of the classes and suffer from a concentration around the origin (i.e., the crowding problem). Our method seems to do a slightly better job at separating the classes in the middle than Laplacian eigenmaps. The embeddings provided by both t-SNE and LargeVis are of much greater quality with respect to class separation even if they leave outliers. Also, both methods find 11 clusters instead of 10 as they split one class into two clusters.

In Table 4 we report the time needed to compute the embeddings using the methods above on the two data sets. The timing of t-SNE and LargeVis on LiveJournal is based on adjusted reported results from [31]. On MNIST, our method has the lowest CPU time, closely followed by Laplacian eigenmaps. Our method is one order of magnitude faster than t-SNE, which is twice slower than LargeVis. On LiveJournal, our method is still the fastest and one order of magnitude faster than t-SNE. LargeVis, while being slower than our method performs rather well. Laplacian eigenmaps exceeded the available memory and did not complete.

From these results we can say that our method is valid for visualization but cannot achieve a quality close to state-of-the-art methods such as t-SNE or LargeVis. However, it has the advantage to be fast and scales well even using a non-optimized mono-thread implementation.

(a) Laplacian eigenmaps

(b) Our method

(c) t-SNE

(d) LargeVis

Figure 5: Visualizations of MNIST using (a) Laplacian eigenmaps, (b) our method, (c) t-SNE (with Barnes-Hut implementation) and (d) LargeVis. The colors correspond to the different categories (i.e., numbers from 0 to 9).

Time [h]	Eigenmaps	t-SNE	LargeVis	Our
MNIST	0.06	0.46	0.26	**0.04**
LiveJournal	-	78.79	10.37	**5.80**

Table 4: 2D Embedding computation time. The default implementation of LargeVis uses parallelism. The value for Eigenmaps on LiveJournal is not reported because it exceeded the maximum memory available (128 GB).

6 Conclusion

In this contribution, we have presented a theoretical way to recover exactly the set of k smallest eigenvectors of a graph Laplacian. We have shown an accelerated algorithm for the approximation of the eigenspace of the Laplacian $\mathcal{L}$ solely based on Gaussian random signals filtering. We proved the bound on the number of signals to be as tight as ever possible. In addition, we proposed an accelerated eigenvalue estimation algorithm based on eigencount techniques. We presented different applications and compared the efficiency against the state of the art, showing the ability for our method to scale with very large N.

This is an interesting result for the field of graph signal processing and many further questions arise in this context. Among them, the design of the filter could be reconsidered. Could we gain even more efficiency by using a naturally polynomial function for the filter instead of the approximation of an ideal low-pass filter? We suggest using exponentially decreasing kernels, which are low-pass and infinitely differentiable and will assign to the eigenvalues an energy proportional to its position in the spectrum. One could wonder whether such design

could allow stopping the computation before the SVD step.

[Properties of projected Gaussians] We stated in section 3.1 that a Gaussian random matrix projected over a basis keeps its Gaussian properties. We will demonstrate the different properties in this appendix.

Let $\mathbf{U} \in \mathbb{R}^{N\times N}$ describe a basis of N orthonormal vectors and $\mathbf{R} \in \mathbb{R}^{N\times d}$ be a Gaussian random matrix with i.i.d. entries $\sim \mathcal{N}(0, \sigma^2)$.

Mathematically,

$$\forall i, j : (\mathbf{U}\,\mathbf{R})_{i,j} = \langle u_{i-1}, r_j \rangle = \sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}, \tag{10}$$

is a linear transformation of the elements of $\mathbf{R}$. Thus, there are Gaussians. Moreover, we already knew that the size of the product is a $N \times d$ matrix. Next, we will evaluate the two first moments of all those entries.

$$\mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right] = \sum_{\ell=1}^{N} u_{i-1}(\ell)\mathbb{E}[r_{\ell,j}] = 0 \tag{11}$$

$$\begin{aligned}\operatorname{Var}\left(\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right) &= \sum_{\ell=1}^{N} u_{i-1}^2(\ell)\operatorname{Var}(r_{\ell,j}) \\ &= \sigma^2 \sum_{\ell=1}^{N} u_{i-1}^2(\ell) = \sigma^2 \end{aligned} \tag{12}$$

This shows that all entries of $\mathbf{U}\,\mathbf{R}$ are identically distributed. Then we can compute the covariance between any two entries $\left((\mathbf{U}\,\mathbf{R})_{i,j}$ and $(\mathbf{U}\,\mathbf{R})_{n,m}\right)$ to ensure independance:

$$\begin{aligned}\operatorname{Cov}(\mathbf{U}\,\mathbf{R}) &= \mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j} \sum_{\ell'=1}^{N} u_{n-1}(\ell') r_{\ell',m}\right] \\ &= \sum_{\ell=1}^{N}\sum_{\ell'=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell')\mathbb{E}[r_{\ell,j} r_{\ell',m}] \\ &= \mathbb{1}_{\{m=j\}} \sum_{\ell=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell)\mathbb{E}[r_{\ell,m}^2] \\ &= \sigma^2 \mathbb{1}_{\{m=j\}} \langle u_{i-1}, u_{n-1}\rangle \\ &= \sigma^2 \mathbb{1}_{\{m=j\}} \mathbb{1}_{\{n=i\}}, \end{aligned} \tag{13}$$

which shows that any two entries in $\mathbf{U}\,\mathbf{R}$ are independant. Combining the last two shows that the entries of $\mathbf{U}\,\mathbf{R}$ are i.i.d. Gaussian random samples with pdf $\sim \mathcal{N}(0, \sigma^2)$ just like $\mathbf{R}$.

Acknowledgment

We would like to thank Dr. O. Lévêque for our insightful discussions on eigenvalues distributions. We are also thankful to Dr. A. Loukas and Prof. P. Vandergheynst for their precious advices regarding this work.

References

[1] A. Y. Ng, M. I. Jordan, Y. Weiss, *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering.," in *NIPS*, vol. 14, pp. 585–591, 2001.

[4] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[5] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, pp. 833–840, 2002.

[6] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the solution of algebraic eigenvalue problems: a practical guide*, vol. 11. Siam, 2000.

[7] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quarterly of applied mathematics*, vol. 9, no. 1, pp. 17–29, 1951.

[8] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[9] D. C. Sorensen, "Implicit application of polynomial filters in ak-step arnoldi method," *Siam journal on matrix analysis and applications*, vol. 13, no. 1, pp. 357–385, 1992.

[10] D. Calvetti, L. Reichel, and D. C. Sorensen, "An implicitly restarted lanczos method for large symmetric eigenvalue problems," *Electronic Transactions on Numerical Analysis*, vol. 2, no. 1, p. 21, 1994.

[11] P. T. Peter Tang and E. Polizzi, "Feast as a subspace iteration eigensolver accelerated by approximate spectral projection," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 354–390, 2014.

[12] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 687–717, 2014.

[13] M. W. Mahoney and L. Orecchia, "Implementing regularization implicitly via approximate eigenvector computation," *arXiv preprint arXiv:1010.0703*, 2010.

[14] Y. Bai, "High performance parallel approximate eigensolver for real symmetric matrices," 2005.

[15] D. Ramasamy and U. Madhow, "Compressive spectral embedding: sidestepping the svd," in *Advances in Neural Information Processing Systems*, pp. 550–558, 2015.

[16] C. Boutsidis, A. Gittens, and P. Kambadur, "Spectral clustering via the power method-provably," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2015.

[17] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, "Compressive spectral clustering," in *33rd International Conference on Machine Learning*, 2016.

[18] F. R. Chung, *Spectral graph theory*, vol. 92. AMS Bookstore, 1997.

[19] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *arXiv preprint arXiv:1307.5708*, 2013.

[20] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[21] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst, "Accelerated filtering on graphs using lanczos method," *arXiv preprint arXiv:1509.04537*, 2015.

[22] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47–68, 2011.

[23] V. A. Marchenko and L. A. Pastur, "Distribution of eigenvalues for some sets of random matrices," *Matematicheskii Sbornik*, vol. 114, no. 4, pp. 507–536, 1967.

[24] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," *arXiv preprint arXiv:1011.3027*, 2010.

[25] E. Di Napoli, E. Polizzi, and Y. Saad, "Efficient estimation of eigenvalue counts in an interval," *Numerical Linear Algebra with Applications*, 2016.

[26] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[27] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[28] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[29] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[30] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[31] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*, pp. 287–297, International World Wide Web Conferences Steering Committee, 2016.